Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2433254 A1 2004/12/25

(21) 2 433 254

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2003/06/25

(41) Mise à la disp. pub./Open to Public Insp.: 2004/12/25

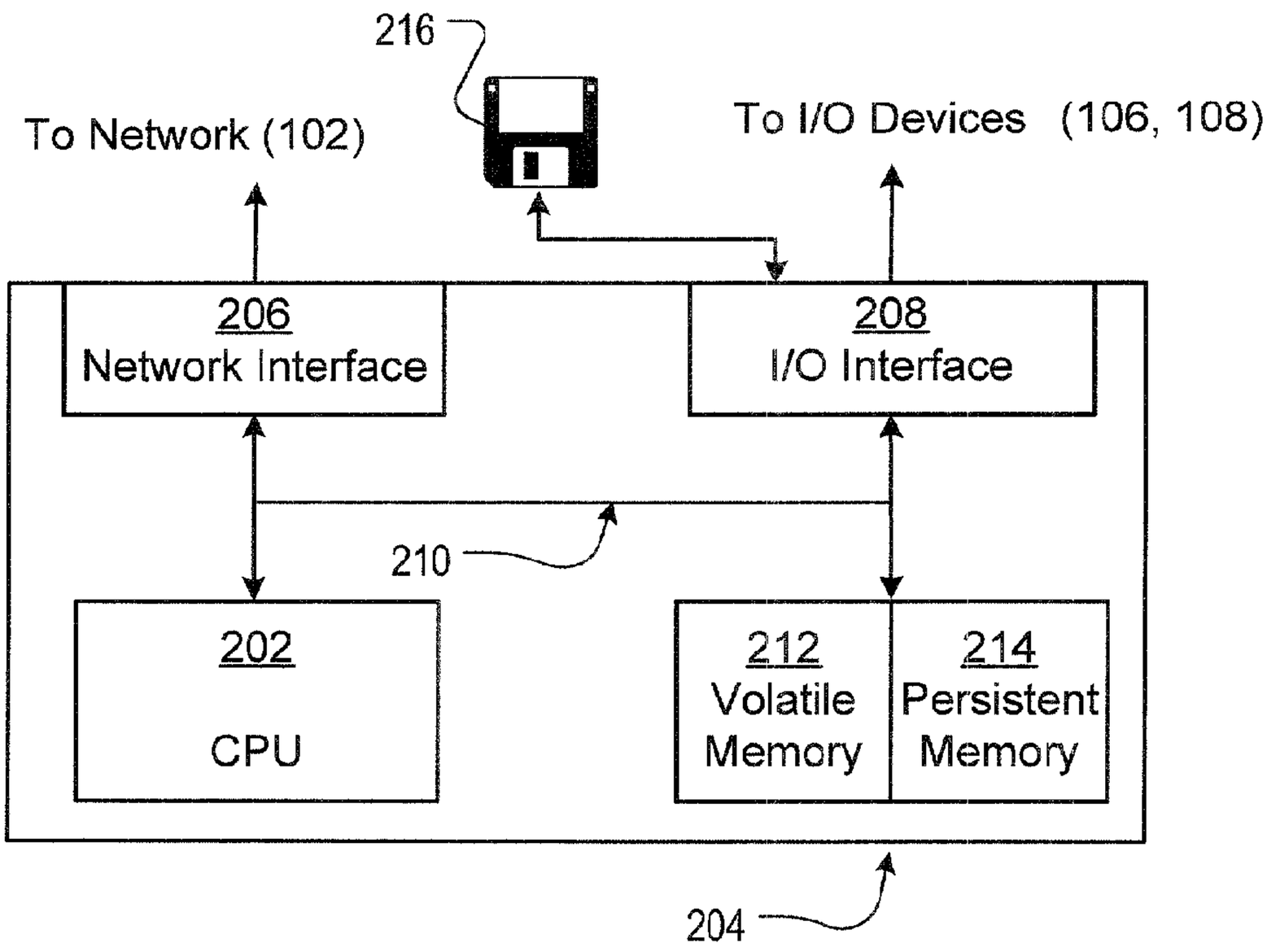
(51) Cl.Int.^{7}/Int.Cl.^{7} G06F 12/02, G06F 13/38, G06F 17/30, G11C 7/10

(71) Demandeur/Applicant:
IBM CANADA LIMITED - IBM CANADA LIMITEE, CA

(72) Inventeurs/Inventors:
SACHEDINA, AAMER, CA;
HURAS, MATTHEW A., CA;
ROMANUFA, KERILEY K., CA

(74) Agent: ROSEN, ARNOLD

(54) Titre : SYSTEME ET METHODE D'ARRET ET DE REDEMARRAGE AUTOMATIQUES D'UN POOL TAMPON
(54) Title: SYSTEM AND METHOD FOR WARM SHUTDOWN AND RESTART OF A BUFFER POOL



(57) Abrégé/Abstract:
A system and method for warm shutdown and restart of a buffer pool is provided. The buffer content including pages and control structures associated with these pages is partitioned into blocks and each block is saved to disk as a sequential file. The size of each block may be selected to provide the optimal I/O (input/output) efficiency during buffer pool shutdown and restart operations. Blocks can be saved simultaneously using a number of writer threads. During restart of the buffer pool, the control information is read from the disk and restored to the buffer pool using reader threads. The buffer pool may be made available for references to pages while reader threads are enabled to read the saved information by blocks to facilitate a more rapid restoration of information in the buffer pool. Allocation of portions of the buffer pool for restoring pages may be performed while the buffer pool is available for references to hasten availability.

ABSTRACT

A system and method for warm shutdown and restart of a buffer pool is provided. The buffer content including pages and control structures associated with these pages is partitioned into blocks and each block is saved to disk as a sequential file. The size of each block may be selected to provide the optimal I/O (input/output) efficiency during buffer pool shutdown and restart operations. Blocks can be saved simultaneously using a number of writer threads. During restart of the buffer pool, the control information is read from the disk and restored to the buffer pool using reader threads. The buffer pool may be made available for references to pages while reader threads are enabled to read the saved information by blocks to facilitate a more rapid restoration of information in the buffer pool. Allocation of portions of the buffer pool for restoring pages may be performed while the buffer pool is available for references to hasten availability.

SYSTEM AND METHOD FOR WARM SHUTDOWN AND RESTART OF A BUFFER POOL

TECHNICAL FIELD

[0001] This system and method are related to the field of database management systems and more particularly to buffer pool shutdown and restart.

BACKGROUND

[0002] Many software systems such as information retrieval systems, database engines and database management systems (DBMSs) use a buffer pool or buffer cache to store recently accessed data. In these systems, buffer pool sizes are relatively large. Capacities in the 100 GB range are common. A buffer pool in such software systems consists of a number of individual fixed sized pages, so that the DBMS manages data in the database and the buffer pool by these pages.

[0003] As the database is referenced during processing of user requests, pages of the database are read from one or more disks storing the database and are cached to the buffer pool once the data in a page becomes accessed. The buffer pool may contain "clean" pages which have not been modified to memory after having been read from disk and may contain "dirty" pages which include modifications to the database in the buffer pool. When the buffer pool is shutdown, dirty pages (that is, data contained in the dirty pages) must be written to disk or other persistent storage in order to maintain the data modifications contained in those pages. Typically, existing software is faced with two problems associated with shutdown and startup of buffer pools. The first problem arises because a buffer pool is managed at the granularity of a page: the pool consists of pages in memory that are most likely not from contiguous disk locations. As a result, when a large percentage of pages are dirty, saving such pages is inefficient because saving dirty pages to disk may require writing to random or non-sequential offsets, making more frequent disk subsystem I/O effort, and the pages in the buffer pool may become saved in a non-contiguous fashion to the persistent disk storage.

[0004] The second problem results from a loss of information when the buffer pool is shutdown. A populated buffer pool contains implicit information by virtue of the pages that are cached to

the buffer pool at any point in time. The set of pages that are cached in the buffer pool at some point in time represents the set of pages that the DBMS considers most likely to be reused and thus most worthy of caching at that point in time. When a buffer pool is shutdown, this cached information may be lost which is highly undesirable.

[0005] When restarted, a buffer pool management sub-system takes time to relearn which pages are the most worthy of caching to the buffer pool. This relearning effort may take a significant amount of time. As a result, the first accesses to the database will be penalized as it is necessary to read the referenced pages from disk rather than from the buffer pool. Thus, the application that needs the data will have to wait longer in comparison to the situation in which the buffer pool had already cached the desired page from persistent storage.

[0006] A third problem common to buffer pool starts and restarts is buffer pool allocation. Typically, a buffer pool is not made available for storing pages to or retrieving pages from the buffer pool until the portion of the buffer pool configured for storing pages is completely allocated in memory. This allocation may unnecessarily delay the start (or restart) of the information retrieval system using the buffer pool.

[0007] A solution to some or all of these shortcomings is therefore desired.

SUMMARY

[0008] The present invention is directed to a system and method for warm buffer pool shutdown and restart.

[0009] In accordance with an aspect the invention, for an information retrieval system coupled to a memory and coupled to a buffer pool maintaining a plurality of pages of recently accessed information, a method for restarting the buffer pool after shutting down the information retrieval system, the method comprising before shutting down the information retrieval system, storing one or more said pages to the memory, and, storing control information associated with said buffer pool to the memory, for use by the information retrieval system to restore the one or more said pages to the buffer pool upon a restart of the information retrieval system.

[0010] In accordance with a feature of this aspect, the memory is persistent memory and the buffer pool is volatile memory.

[0011] In accordance with another feature of the invention, the information retrieval system restores the buffer pool with the one or more of said pages upon a restart of the information retrieval system.

[0012] Storing the one or more pages may comprise sequentially storing blocks of pages where each block comprises one of a page and a group of sequential pages. A size of each of said blocks is preferably selected to optimize input/output operations. Further, storing of blocks may be in parallel.

[0013] A feature provides that restoring said control information and pages comprises reading blocks of pages where each block comprises one of a page and group of sequential pages. A size of each of said blocks is preferably selected to optimize input/output operations. Further, reading of blocks may be in parallel.

[0014] In accordance with another feature of this aspect, following the restoring of the control information, reference to pages of the buffer pool is permitted while restoring said pages. Permitting reference may comprise latching a particular page to delay a reading of the particular page from said buffer pool until said particular page is restored from said persistent memory. Further, permitting reference to pages of the buffer pool while restoring said pages may comprise allocating portions of the buffer pool for storing said pages. As well, while restoring said pages, the restoring of a particular page in response to a reference to the particular page may be prioritized.

[0015] To restore blocks, read requests instructing the reading of said blocks may be generated so that reading the blocks is performed in accordance with the read requests. Generating read requests may comprise ordering the requests in response to a predicted relative need for each of the blocks.

[0016] In accordance with a further feature, the memory may comprise a hard disk drive.

[0017] In accordance with another aspect of the invention, there is provided, for an information retrieval system having a memory and a buffer pool maintaining a plurality of pages of recently accessed information for re-access, a restart system comprising one or more writers adapted to, in response to a buffer pool shutdown, store one or more said pages to the memory and store control information associated with said buffer pool to the memory for restoring the one or more pages to the buffer pool upon a restart.

[0018] In accordance with one feature of the invention, The restart system comprises one or more readers adapted to restore said control information and pages to enable the buffer pool for re-access in response to a buffer pool restart.

[0019] In yet a further aspect of the invention, there is provided, for an information retrieval system coupled to a memory and coupled to a buffer pool maintaining a plurality of pages of recently accessed information, a computer program product having a computer readable medium tangibly embodying computer executable code for directing the information retrieval system to restart the buffer pool after shutting down the information retrieval system, the computer program product comprising code for storing one or more said pages to the memory before shutting down the information retrieval system, and, code for storing control information associated with said buffer pool to the memory, the control information for use by the information retrieval system to restore the one or more said pages to the buffer pool upon a restart of the information retrieval system.

[0020] In accordance with one feature of the invention, the computer program product comprises code for, on a restart, restoring said control information and pages to enable the buffer pool for re-access.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] Further features and advantages of aspects of the present invention will become apparent from the following detailed description, taken in combination with the appended drawings, in which:

[0022] Fig. 1 schematically illustrates a computer system embodying aspects of the invention;

[0023] Fig. 2 schematically illustrates, in greater detail, a portion of the computer system of Fig. 1;

[0024] Fig. 3 illustrates, in functional block form, a portion of the memory illustrated in Fig. 2;

[0025] Fig. 4 schematically illustrates a buffer pool as illustrated in Fig. 3;

[0026] Fig. 5 is a flowchart illustrating basic operational steps involved in warm shutdown of the buffer pool illustrated in Fig. 3; and

[0027] Fig. 6 is a flowchart illustrating basic operational steps involved in restarting of the buffer pool;

[0028] It will be noted that throughout the appended drawings, like features are identified by like reference numerals.

DETAILED DESCRIPTION

[0029] The following detailed description of the embodiments of the present invention does not limit the implementation of the invention to any particular computer programming language. The present invention may be implemented in any computer programming language provided that the OS (Operating System) provides the facilities that may support the requirements of the present invention. A preferred embodiment is implemented in the C or C++ computer programming language (or other computer programming languages in conjunction with C/C++). Any limitations presented would be a result of a particular type of operating system, data processing system, or computer programming language, and thus would not be a limitation of the present invention.

[0030] An embodiment of the invention, computer system 100, is illustrated in FIG.1. Computer system 100, which is illustrated for exemplary purposes as a computing device, is adapted to communicate with other computing devices (not shown) using network 102. As will be appreciated by those of ordinary skill in the art, network 102 may be embodied using conventional networking technologies and may include one or more of the following: local networks, wide area networks, intranets, the Internet, and the like.

[0031] Through the description herein, an embodiment of the invention is illustrated with aspects of the invention embodied solely on computer system 100. As will be appreciated by those of ordinary skill in the art, aspects of the invention may be distributed amongst one or more networked computing devices which interact with computer system 100 using one or more networks such as, for example, network 102. However, for ease of understanding, aspects of the invention have been embodied in a single computing device – computer system 100.

[0032] Computing device 100 typically includes a processing system 104 which is enabled to communicate with the network 102, and various input devices 106 and output devices 108. Input devices 106, (a keyboard and a mouse are shown) may also include a scanner, an imaging system (e.g., a camera, etc.), or the like. Similarly, output devices 108 (only a display is illustrated) may also include printers and the like. Additionally, combination input/output (I/O) devices may also be in communication with processing system 104. Examples of conventional I/O devices (not shown in FIG.1) include removable recordable media (e.g., floppy disk drives, tape drives, CD-ROM drives, DVD-RW drives, etc.), touch screen displays, and the like.

[0033] Exemplary processing system 104 is illustrated in greater details in FIG. 2. As illustrated, processing system 104 includes: a central processing unit (CPU) 202, memory 204, network interface (I/F) 206 and input-output interface (I/O I/F) 208. Communication between various components of the processing system 104 may be facilitated via a suitable communications bus 210 as required.

[0034] CPU 202 is a processing unit, such as an Intel Pentium™, IBM PowerPC™, Sun Microsystems UltraSparc™ processor, or the like, suitable for the operations described herein. As will be appreciated by those of ordinary skill in the art, other embodiments of processing system 104 could use alternative CPUs and may include embodiments in which two or more CPUs are employed (not shown). CPU 202 may include various support circuits to enable communication between itself and the other components of processing system 104.

[0035] Memory 204 includes both volatile memory 212 and persistent memory 214 for the storage of: operational instructions for execution by CPU 202, data registers, application and thread storage, and the like. Memory 204 preferably includes a combination of random access

memory (RAM), read only memory (ROM) and persistent memory such as that provided by a hard disk drive, flash memory or the like.

[0036] Network I/F 206 enables communication between other computing devices (not shown) and other network computing devices via network 102. Network I/F 206 may be embodied in one or more conventional communication devices. Examples of a conventional communication device include: an Ethernet card, a token ring card, a modem, or the like. Network I/F 206 may also enable the retrieval or transmission of instructions for execution by CPU 202, from or to a remote storage media or device via network 102.

[0037] I/O I/F 208 enables communication between processing system 104 and the various I/O devices 106 and 108. I/O I/F 208 may include, for example, a video card for interfacing with an external display such as output device 108. Additionally, I/O I/F 208 may enable communication between processing system 104 and a removable media 216. Removable media 216 may comprise a conventional diskette or other removable memory devices such as Zip™ drives, flash cards, CD-ROMs, static memory devices and, the like. Removable media 216 may be used to provide instructions for execution by CPU 202 or as a removable data storage device.

[0038] Computer instructions/applications stored in memory 204 and executed by CPU 202 (thus adapting the operation of the computer system 100 as described herein) are illustrated in functional block form in FIG. 3. As will be appreciated by those of ordinary skill in the art, the discrimination between aspects of the applications illustrated as functional blocks in FIG. 3 is somewhat arbitrary in that the various operations attributed to a particular application as described herein may, in an alternative embodiment, be subsumed by another application.

[0039] As illustrated for exemplary purposes only, memory 204 stores instructions and data for enabling the operation of the system for buffer pool warm shutdown and restart that include: an operating system (OS) 302, a communication suite 304, a database management system (DBMS) 306 adapted to include a warm shutdown and restart function 308 and a buffer pool 316. Warm shutdown and restart function 308 comprises in the exemplary embodiment, one or more writer threads 312 (multiple writer threads 312a, 312b, 312i are shown), one or more reader threads 314 (multiple reader threads 314a, 314b, 314j are shown) and a restore control 315. It will be apparent to persons skilled in the art that the buffer pool is typically stored in volatile memory

212 when in operation while the remaining instructions and data may also be stored to persistent memory during operation.

[0040] The applications such as OS 302, communications suite 304, DBMS 306, the shutdown and restart application 308, writer threads 312 and reader threads 314 may be stored to the volatile memory 214 and/or persistent memory 214. The buffer pool 316 is stored in the volatile memory 212.

[0041] OS 302 is an operating system suitable for operation with a selected CPU 202 and the operations described herein. Multi-tasking, multi-threaded OSes such as, for example, IBM AIX™, Microsoft Windows, Linux or the like, are expected to be preferred in many embodiments. Buffer 316 is specifically reserved as a region of the memory 204 for storing information. Though shown as a single contiguous region, persons skilled in he art will understand that buffer pool 316 may comprise two or more regions of memory 212. Typically, each region is a multiple of the page size used by DBMS 306. Communication suite 304 provides, through interaction with OS 302 and network I/F 206 (FIG. 2), suitable communication protocols to enable communication with other networked computing devices via network 102 (FIG. 1). Communication suite 304 may include one or more of such protocols such as TCP/IP, Ethernet, token ring and the like. Communications suite 304 preferably includes asynchronous transport communication capabilities for communicating with other computing devices.

[0042] DBMS 306 is a software application executable in the environment of OS 302 in order to provide information stored in a database (none shown) to client applications (not shown) on request. The database is typically stored in persistent memory 214 or other persistent storage coupled to computer system 100. Client applications also can be run in the environment of OS 302 or can be run on other computer systems adapted to communicate with computer system 100 using the network 102.

[0043] DBMS 306, in order to minimize disk access and/or network traffic flow, stores recently accessed data from the database to the buffer pool 316 in volatile memory 212. The buffer pool 316 is often set up as part of DBMS 306 work space.

[0044] DBMS 306 may require being shutdown and restarted. In accordance with in the prior art, DBMSs save only dirty pages stored in the buffer pool 316 to the persistent storage storing the database requiring a plurality of shutdown or non-sequential writes. In accordance with the invention, DBMS 306 as adapted for warm shutdown, saves pages (for example, but not limited to, dirty pages) of the buffer pool 316 and control information associated with the pages to sequential space on persistent memory 214 using writer threads 312. Persistent memory 214 for such purposes typically comprises a hard disk drive but may include flash memory or another persistent storage device that provides sufficient I/O characteristics for fast shutdowns and restores. For example, tape media or writable CD-ROMs, DVDs or the like generally have slow write response times. Persistent storage devices with I/O characteristics similar to hard disk drives are preferred. Buffer pool information may be written in a parallel way whereby writer threads 312 can write different portions of the buffer pool at the same time to contiguous space (for example using an OS primitive such as vectored write). For I/O efficiency, large block writes such as several megabytes per write as opposed to a single page (few thousand bytes) per write may be preferred. In such a case of large block writes, both clean and dirty pages are typically written out. In order to restore data in the buffer pool 316, DBMS 306 employs reader threads 314 for restoring saved data from persistent memory 214 to buffer pool 316 as described further herein. While restoring saved data from persistent memory 214, the restore control 315 permits reference to the saved data.

[0045] Fig. 4 illustrates schematically an exemplary structure of the buffer pool 316. The buffer pool 316 is used to store recently accessed data, which is stored as a plurality of memory pages 402, and control structures 404. The control structures 404 store information for describing a state of each individual page 402. Typically control structures comprise but are not limited to, page descriptor directory (PDD) 406 and log control block (LCB) 408.

[0046] Each individual page of the pages 402 may be clean (schematically illustrated as a white rectangle 410) or dirty comprising data in the buffer pool 316 that has not been stored to the persistent data store for the database (schematically illustrated as a black rectangle 412). Buffer pool 316 comprises one or more contiguous regions of volatile memory 212 where the clean and dirty pages are not separated from each other. In order to differentiate clean and dirty pages, information is maintained in an assigned page descriptor (PD) stored in PDD 406. The PD also

comprises a page latch construct that is used to protect access to the page and a log sequence number (LSN) for maintaining information indicating whether the page is modified. In LCB 408, DBMS 306 maintains information such as the minimum LSN representing the oldest modification to one of the buffer pool pages in order to control how far back the DBMS 306 must maintain log records in the primary (as opposed to the archived) log space needed for crash recovery purposes.

[0047] That portion 402 of the buffer pool 316 where pages are maintained may be logically divided into a number of blocks (three blocks 414A, 414B and 414k are shown for illustrative purposes only). The size of each block may be selected to optimize input/output (I/O) efficiency (i.e. disk read and disk write access efficiency) during buffer pool shutdown and restart operations. Thus a block may be a single page or a group of contiguous pages. DBMS 306 uses writer threads 312A, 312B and 312i and may also make the process of saving information from the buffer pool 316 more effective using parallel writing of blocks 414A, 414B and 414k. Upon initiation of warm shutdown, each writer thread 312A, 312B and 312i saves a respective assigned block 414A, 414B and 414k at the same time. The same multi-thread principles may also be applied to restoring operations when the saved information is read in and restored to buffer pool 316 by a number of reader threads 314 that may read and restore information to the buffer pool simultaneously. Though described in terms of writing out blocks 414, it is understood that each writer thread 312 may be directed to write only dirty pages rather than writing the entire buffer pool.

[0048] The warm shutdown and restart of the buffer is performed in two stages. The warm shutdown stage is described in greater detail below with reference to Fig. 5, and the restart stage is described in greater detail below with reference to Fig. 6.

[0049] Fig. 5 is a flowchart illustrating the main operational steps involved in warm shutdown of the buffer pool 316 in an exemplary embodiment. On receipt of a warm shutdown request from DBMS 306 (Step 500), warm shutdown and restart function 308 saves information stored in the buffer pool 316 as a sequential disk file. The sequential disk file is saved to a user declared temporary disk or shutdown space. It is common that such temporary space already exists in DBMS 306 for temporary objects such as temporary tables and thus no additional disk space is

necessary. The size of temporary disk space has to be sufficient to accommodate pages 402 stored in the buffer pool 316 and control structures 404. At step 502, the control structures 404 are saved. Pages 402 stored in the buffer pool 316 are saved to the sequential disk file on disk (Step 504). Warm shutdown and restart function 308, in order to minimize the duration of the warm shutdown, may use a number of writer threads 312 that are adapted to perform write operations simultaneously. Each writer thread 312 writes a respectively assigned block (414A, 414B, 414k) of buffer pool memory to the contiguous disk file simultaneously. On completion of writing information from the buffer pool 316 (Step 506), the DBMS 306 or the entire computer system 100 may also be shutdown in accordance with operations that will be understood to persons skilled in the art.

[0050] Fig. 6 is a flowchart illustrating the basic operational steps involved in a restart of the buffer pool 316 in accordance with an embodiment. After restarting DBMS 306, warm shutdown and restart function 308 may be invoked to restart buffer pool 316 in the memory 212. DBMS 306 determines whether to restart the buffer pool 316 as an empty buffer or use the warm shutdown and restart function 308 to restore information that was saved during the warm shutdown stage. On receipt of a restart signal from DBMS 306 (Step 600), the warm shutdown and restart function 308 reads control structure information from the contiguous disk file stored on disk (not shown)(Step 602) and recreates control structure 404 in the buffer pool 316 (Step 604). In order to recreate control structures, page descriptors are recreated in PDD 406. Then, using the information contained in the page descriptors, other control structures such as hash tables for quick lookup, linked lists to chain dirty pages, etc. which are required to support DBMS 306 operations, are also recreated.

[0051] After re-creation of the control structures 404, the latch constructs of the descriptors stored in PPD 406 are invoked to latch (e.g. Xlatch) buffer pages 402 (Step 606) in order to protect read access to those pages. This enables immediate reference to the pages before the content of these pages is restored to the buffer pool 316. After latching the pages, the buffer pool 316 is operable. As is described above, the page portion 402 of the buffer pool 316 has been logically divided into blocks and each of these blocks was saved to the contiguous disk file (step 504, Fig. 5). In order to restore pages to the buffer pool 316, each respective block (414A, 414B and 414k) is restored by reader threads 314. Block read requests are put in queues for service by

reader threads 314 which operate asynchronously with respect to each other. These block read requests provide sequential prefetch to reader threads 314 (Step 610). On receipt of a block read request from the queue (Step 610), one of the reader threads 314 services the request by reading the block of pages from the sequential disk files and restoring same into the buffer pool. Information from the temporary disk store is read on a block by block basis (Step 614). The I/O is sequential and is performed in block sizes to optimize restoring operations. As each block is restored to the buffer pool 316, the pages for that block are unlatched (Step 616). One of the large costs – in terms of time – to start a buffer pool is the allocation of the buffer pool memory (volatile). In an alternative embodiment, the buffer pool 316 may be made available following the allocation and readying of the PDs, which represent a small fraction of the buffer pool size, without allocating the portion of the buffer pool 402 where pages are stored. Further allocation of the volatile memory necessary for the blocks to be restored may be performed by the reader threads before each reads an assigned block from the shutdown space. The actual buffer pool page that a PD represents need not be present when the buffer pool is opened up since the latch on the PD is held and a reader thread will only release it after first allocating the memory and then reading in the block facilitating a speedier database start up. The deferral of the allocation of the page storing portion of the buffer pool may be performed when undertaking a cold start of the buffer pool as well. Persons of ordinary skill in the art will appreciate that reader threads or other prefetchers receiving instructions to fetch pages from a database store or the shutdown space may allocate pages of the buffer pool in response to the demand to read pages for storing to the pool.

[0052] Since the buffer pool 316 is open for use before the pages are actually restored, it is possible that a client may request a page which has yet to be read and restored. The restore control 315 permits reference to pages 402 of the buffer pool 316 while restoring said pages 402. In this case, the client will have to wait on the page latch (since all pages are latched in advance (step 606)). When the page has been restored into the buffer pool 316 by the reader threads 314, the page is unlatched and the client will be granted access to use the page.

[0053] In another embodiment of the invention, warm shutdown and restart function 308 employing the restore control 315 may prioritize the restoring of a particular page in response to a client reference to the particular page. More particularly, the process of restoring pages may be

organized in such a way that when a page which is not yet restored to the buffer pool 316 is first requested, the reading of the page may be advanced out of the order determined by the queued read requests. The page may be read either by the requestor or by reordering the requests to the read threads. In a further alternative, information indicating the relative likelihood that particular page will be required before other pages may be maintained, for example in the PDs. This relative weight indicating hot pages may be used to order the read requests in the queue. For block read requests, an average or other block-wide rating may be computed from the ratings for individual pages. The restart subsystem could then queue up the pre-fetch read requests in a more intelligent fashion thereby predicting which pages are likely to be accessed first by clients.

[0054] The warm shutdown and restore of the buffer pool in accordance with the present invention serves to reduce shutdown and restart durations. I/O efficiency is optimized when saving and restoring the buffer pool's contents including pages and control information associated with the pages. The restart of a pre-populated buffer pool may enhance query processing upon restart.

[0055] The embodiment(s) of the invention described above is(are) intended to be exemplary only. The scope of the invention is therefore intended to be limited solely by the scope of the appended claims.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. For an information retrieval system coupled to a memory and coupled to a buffer pool maintaining a plurality of pages of recently accessed information, a method for restarting the buffer pool after shutting down the information retrieval system, the method comprising:

a) before shutting down the information retrieval system, storing one or more said pages to the memory; and,

b) storing control information associated with said buffer pool to the memory, for use by the information retrieval system to restore the one or more said pages to the buffer pool upon a restart of the information retrieval system.

2. The method claimed in claim 1 wherein the memory is persistent memory and the buffer pool is volatile memory.

3. The method claimed in claim 1 wherein the information retrieval system restores the buffer pool with the one or more pages upon a restart of the information retrieval system.

4. The method claimed in claim 1 wherein storing said one or more pages comprises sequentially storing blocks of pages, each block comprising one of a page and a group of sequential pages.

5. The method claimed in claim 4 wherein a size of each of said blocks is selected to optimize input/output operations.

6. The method claimed in claim 4 comprising storing said blocks in parallel.

7. The method claimed in claim 3 wherein restoring said control information and one or more pages comprises reading blocks of pages, each block comprising one of a page and a group of sequential pages.

8. The method claimed in claim 7 comprising selecting a size of each said block to optimize input/output operations.

9. The method claimed in claim 7 comprising reading blocks in parallel.

10. The method claimed in claim 3 comprising, following the restoring of the control information, permitting reference to pages of the buffer pool while restoring said one or more pages.

11. The method claimed in claim 10 wherein permitting reference comprises latching a particular page to delay a reading of the particular page from said buffer pool until said particular page is restored from said memory.

12. The method claimed in claim 11 wherein permitting reference to pages of the buffer pool while restoring said pages comprises allocating portions of the buffer pool for storing said one or more pages.

13. The method claimed in claim 10 comprising, while restoring said pages, prioritizing the restoring of a particular page in response to a reference to the particular page.

14. The method claimed in claim 7 comprising generating read requests instructing the reading of said blocks and reading the blocks in accordance with the read requests.

15. The method claimed in claim 14 wherein generating read requests comprises ordering the requests in response to a predicted relative need for each of the blocks.

16. The method claimed in claim in 1 wherein said memory comprises a hard disk drive.

17. For an information retrieval system having a memory and a buffer pool maintaining a plurality of pages of recently accessed information for re-access, a restart system comprising:

one or more writers adapted to, in response to a buffer pool shutdown, store one or more said pages to the memory and store control information associated with

said buffer pool to the memory for restoring the one or more pages to the buffer pool upon a restart.

18. The restart system claimed in claim 17 wherein the memory is a persistent memory and the buffer pool is a volatile memory.

19. The restart system claimed in claim 17 comprising:

one or more readers adapted to restore said control information and pages to enable the buffer pool for re-access in response to a buffer pool restart.

20. The restart system claimed in claim 19 wherein to optimize input/output operations said writers write blocks of sequential pages and said readers read blocks of sequential pages.

21. The restart system claimed in claim 20 wherein said writers write said blocks in parallel and wherein said readers read said blocks in parallel.

22. The restart system claimed in claim 19 comprising a restore control to, following the restoring of the control information, permit reference to pages of the buffer pool while restoring said pages.

23. The restart system claimed in claim 22 wherein the restore control is adapted to, while restoring said pages, prioritize the restoring of a particular page in response to a reference to the particular page.

24. For an information retrieval system coupled to a memory and coupled to a buffer pool maintaining a plurality of pages of recently accessed information, a computer program product having a computer readable medium tangibly embodying computer executable code for directing the information retrieval system to restart the buffer pool after shutting down the information retrieval system, the computer program product comprising:

a) code for storing one or more said pages to the memory before shutting down the information retrieval system; and,

b) code for storing control information associated with said buffer pool to the memory, the control information for use by the information retrieval system to restore the one or more said pages to the buffer pool upon a restart of the information retrieval system.

25. The computer program product of claim 24 wherein the memory is a persistent memory and the buffer pool is a volatile memory.

26. The computer program product claimed in claim 24 comprising code for, on a restart of the information retrieval system:

restoring said control information and one or more pages to the buffer pool.

27. The computer program product claimed in claim 24 wherein code for storing said pages comprises code for sequentially storing blocks of pages; and

wherein each block comprising a group of sequential pages selected to optimize input/output operations.

28. The computer program product claimed in claim 27 wherein said code for sequentially storing blocks comprises code for storing said blocks in parallel.

29. The computer program product claimed in claim 26 wherein code for restoring said control information and pages comprises code for reading blocks of pages; and

wherein each block comprising a group of sequential pages selected to optimize input/output operations.

30. The computer program product claimed in claim 29 wherein said code for reading blocks comprises code for reading said blocks in parallel.

31. The computer program product claimed in claim 26 comprising code for, following the restoring of the control information, permitting reference to pages of the buffer pool while restoring said one or more pages.

32. The computer program product claimed in claim 31 wherein said code for permitting reference comprises code for allocating portions of the buffer pool for storing the one or more pages.

33. The computer program product claimed in claim 31 wherein said code for permitting reference comprises code for latching a particular page to delay a reading of the particular page from said buffer pool until said particular page is restored from said memory.

34. The computer program product claimed in claim 31 comprising code for, while restoring said one or more pages, prioritizing the restoring of a particular page in response to a reference to the particular page.

35. The computer program product claimed in claim 29 comprising code for generating read requests instructing the reading of said blocks and code for reading the blocks in accordance with the read requests.

36. The computer program product claimed in claim 35 wherein the code for generating read requests comprises code for ordering the requests in response to a predicted relative need for each of the blocks.

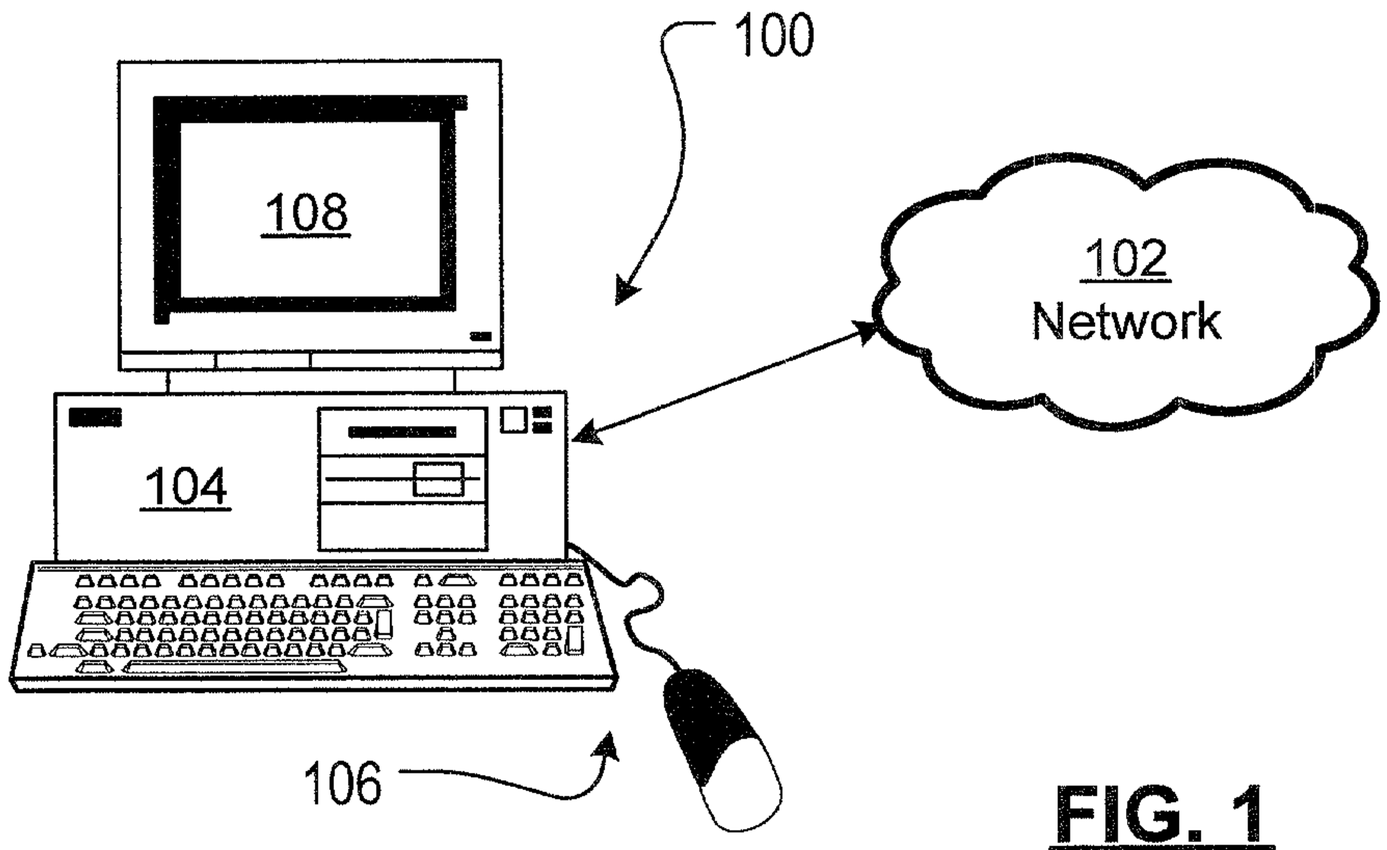

100
108
102
Network
104
106

FIG. 1

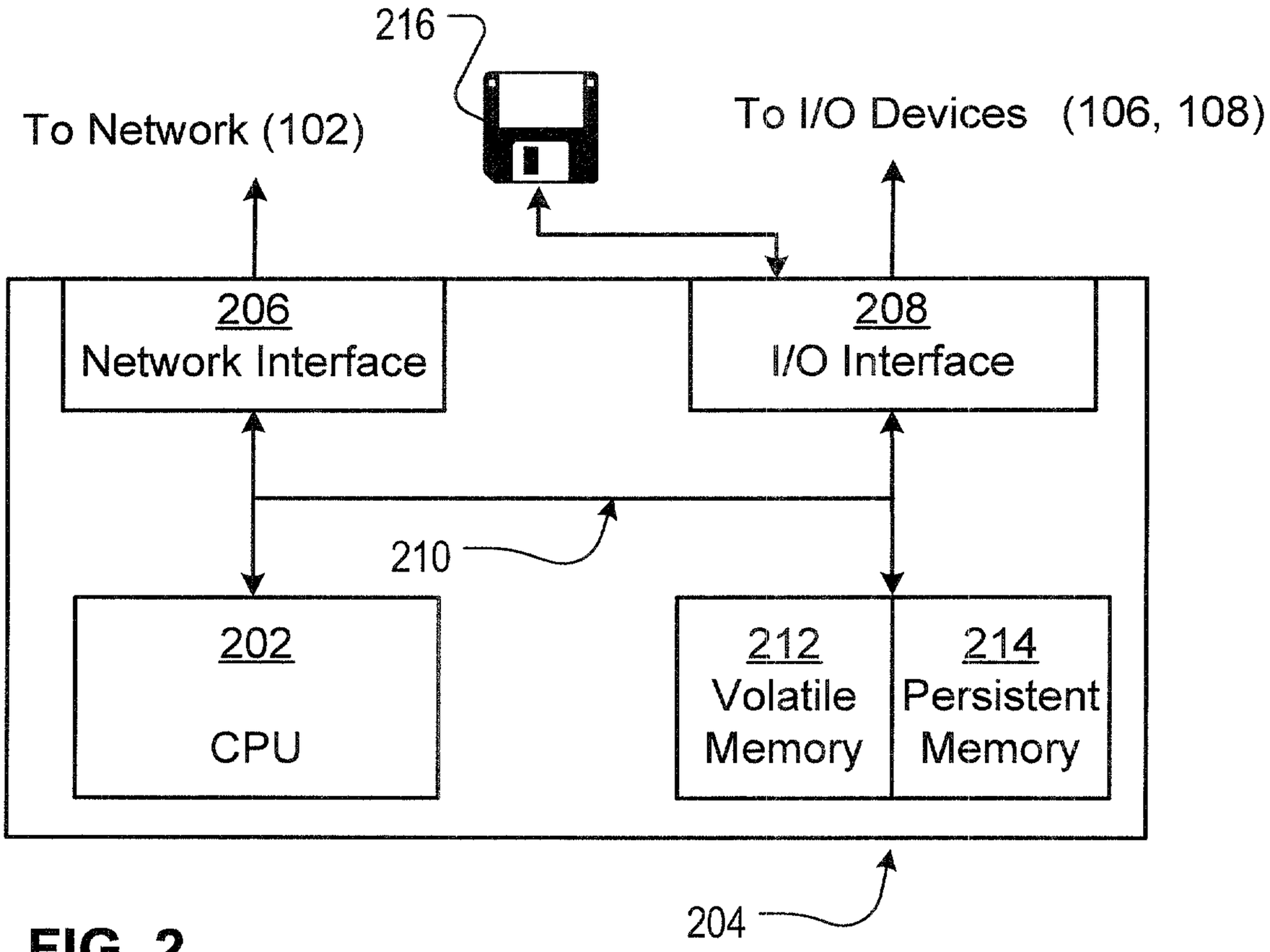

216
To Network (102)
To I/O Devices (106, 108)
206
Network Interface
208
I/O Interface
210
202
CPU
212
Volatile
Memory
214
Persistent
Memory
204

FIG. 2

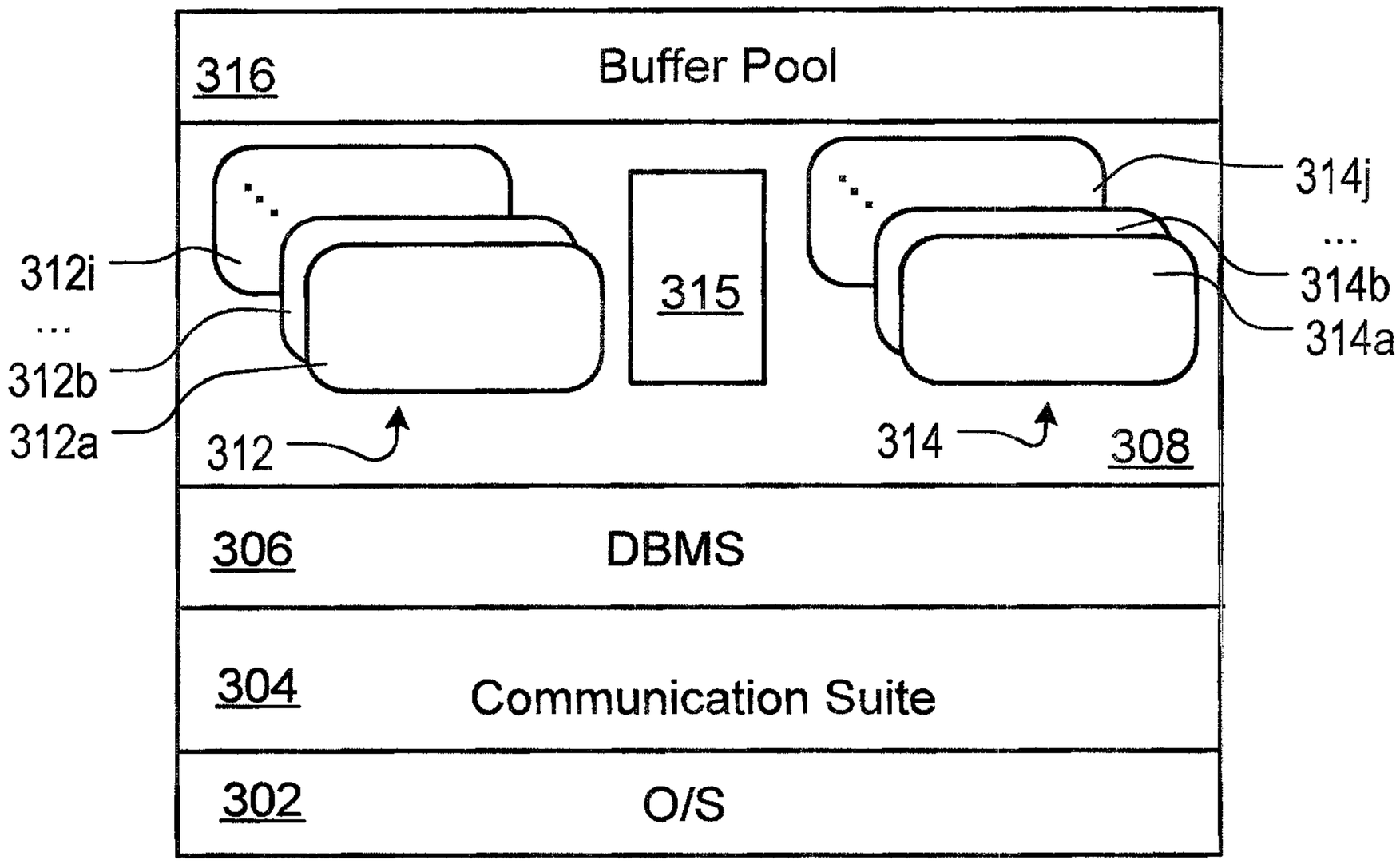

316
Buffer Pool
312i
...
312b
312a
312
315
314j
...
314b
314a
314
308
306
DBMS
304
Communication Suite
302
O/S

FIG. 3

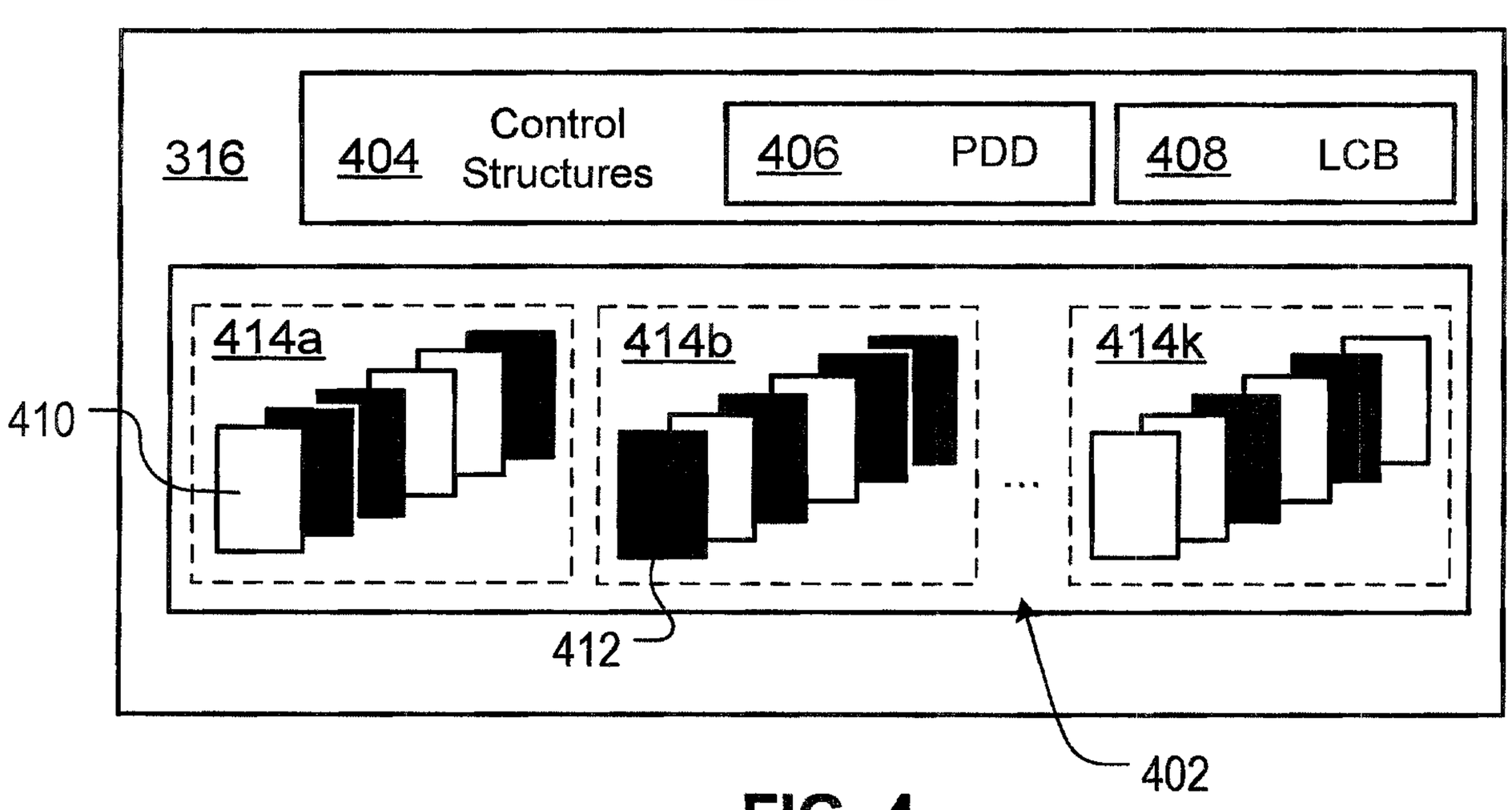

316
404
Control Structures
406
PDD
408
LCB
414a
414b
414k
410
412
402

FIG. 4

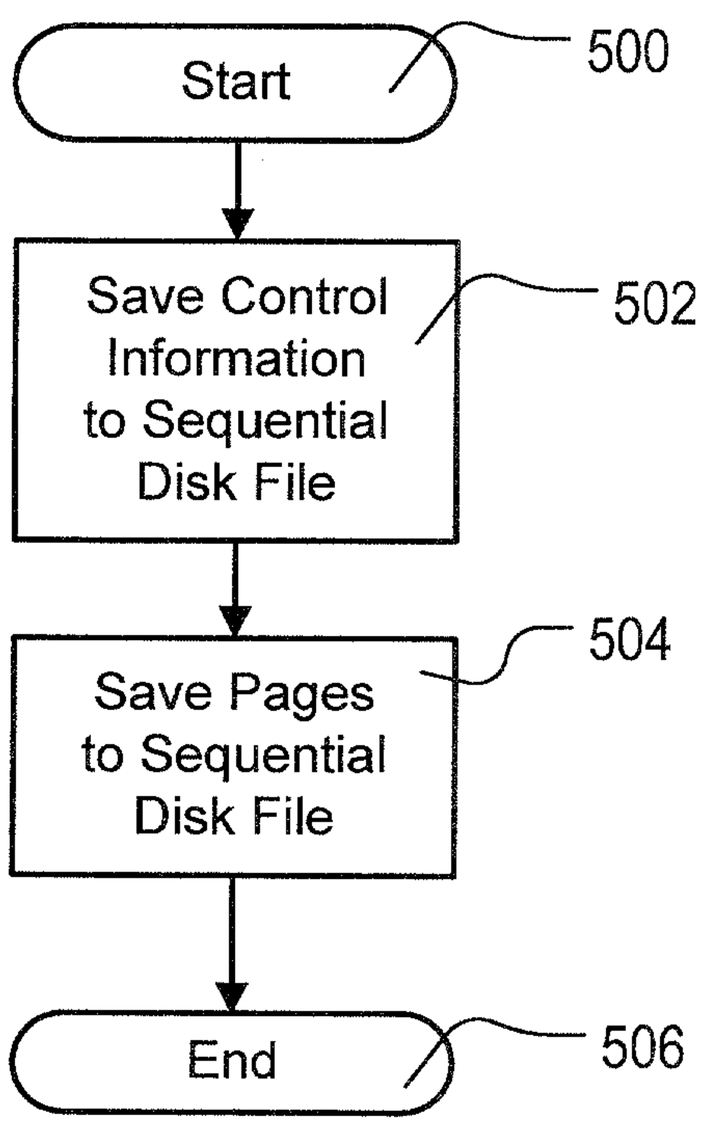

Start
500
Save Control Information to Sequential Disk File
502
Save Pages to Sequential Disk File
504
End
506

FIG. 5

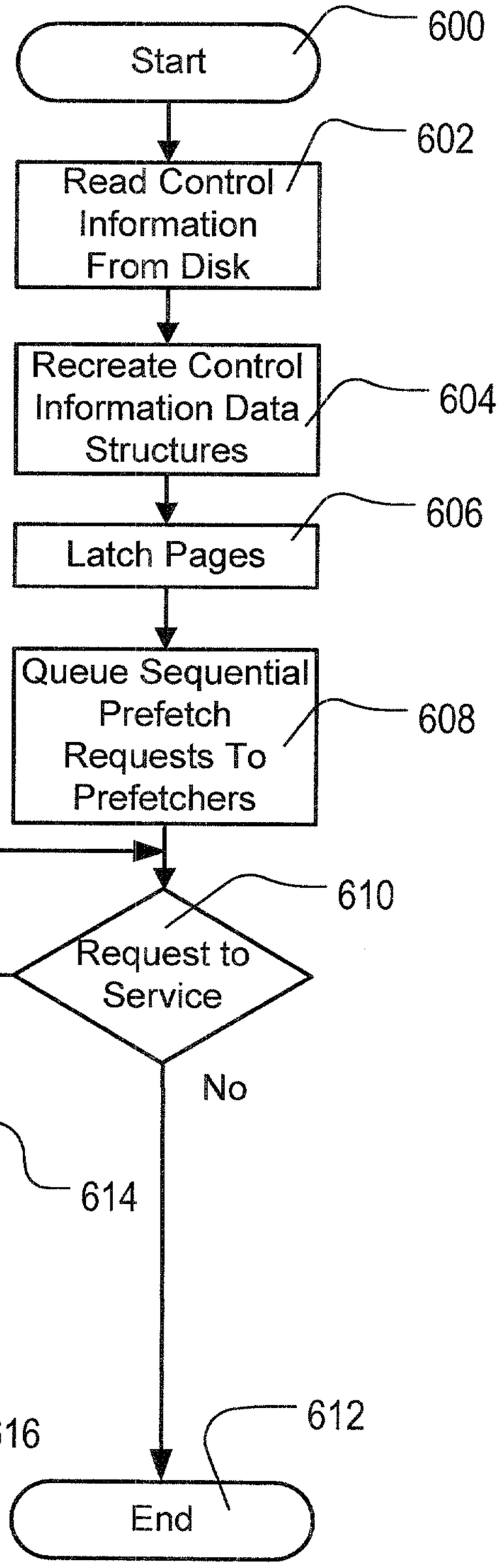
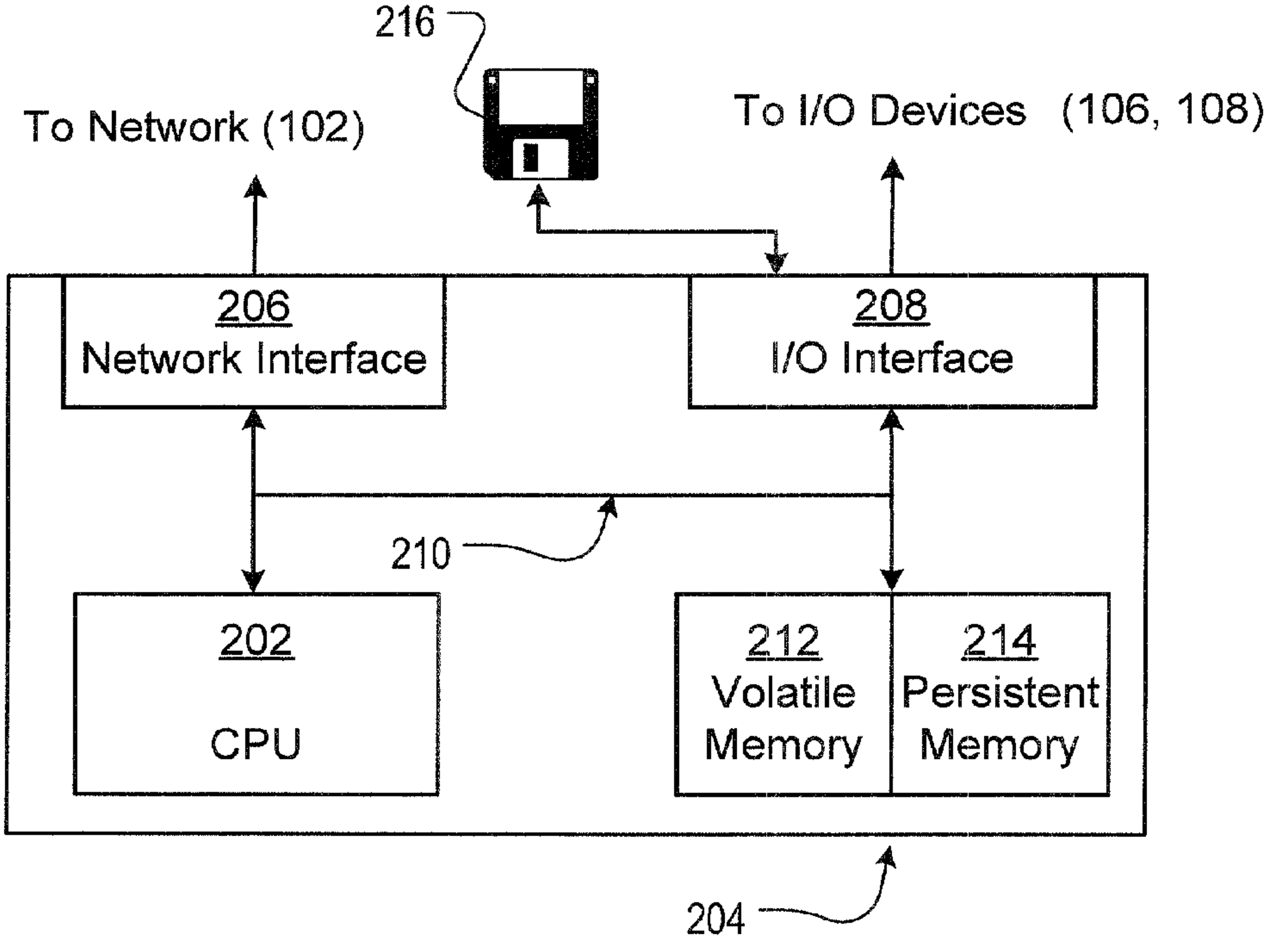

Start
600
Read Control Information From Disk
602
Recreate Control Information Data Structures
604
Latch Pages
606
Queue Sequential Prefetch Requests To Prefetchers
608
Request to Service
610
Yes
No
Service Queue to Restore Pages
614
Unlatch Restored Pages
616
End
612

FIG. 6


216
To Network (102)
To I/O Devices (106, 108)
206
Network Interface
208
I/O Interface
210
202
CPU
212
Volatile
Memory
214
Persistent
Memory
204